Solving a Discrete Logarithm Problem with Auxiliary Input on a 160-bit Elliptic Curve

Yumi Sakemi[1,⋆], Goichiro Hanaoka[2], Tetsuya Izu[1],
Masahiko Takenaka[1], and Masaya Yasuda[1]

[1] FUJITSU LABORATORIES Ltd.,
4-1-1, Kamikodanaka, Nakahara-ku, Kawasaki, 211-8588, Japan
{sakemi,izu,takenaka,myasuda}@labs.fujitsu.com
[2] Research Institute for Secure Systems (RISEC),
National Institute of Advanced Industrial Science and Technology (AIST),
Central 2, 1-1-1, Umezono, Tsukuba, 305-8568, Japan
hanaoka-goichiro@aist.go.jp

Abstract. A discrete logarithm problem with auxiliary input (DLPwAI) is a problem to find α from G, αG, $\alpha^d G$ in an additive cyclic group generated by an element G of prime order r, and a positive integer d satisfying $d|(r-1)$. The infeasibility of this problem assures the security of some cryptographic schemes. In 2006, Cheon proposed a novel algorithm for solving DLPwAI (Cheon's algorithm). This paper reports our experimental results of Cheon's algorithm by implementing it with some speeding-up techniques. In fact, we have succeeded to solve DLPwAI on a pairing-friendly elliptic curve of 160-bit order in 1314 core days. Implications of our experiments on cryptographic schemes are also discussed.

Keywords. DLPwAI (DLP with Auxiliary Input), Barreto-Naehrig pairing-friendly elliptic curve, Cheon's algorithm

1 Introduction

Let $\mathbb{G}$ be an additive cyclic group generated by an element G of prime order r. A discrete logarithm problem (DLP) is a problem to find α from G and αG. In the general setting, DLP is considered to be infeasible, and the infeasibility of DLP assures the security of some cryptographic schemes such as ECDH and ECDSA. When $\mathbb{G}$ is defined on elliptic curves over finite fields, the currently best algorithms for solving DLP require exponential time with regard to r, namely, $O(\sqrt{r})$. In fact, Shanks' baby-step giant-step (BSGS) method [20] requires $O(\sqrt{r})$ group operations in time and $O(\sqrt{r})$ group elements in space. On the other hand, Pollard's ρ-method also requires $O(\sqrt{r})$ in time, but much smaller elements in space. Since the state-of-the-art record of solving DLP on elliptic curves is 112-bit [7], 160-bit elliptic curves have been used as a secure parameter.

⋆ This research was done while she was a doctor student in Okayama University.

Table 1. Required time for solving DLPwAI

	$\log_2 r$ (in bit)	Required Time (by a single core)	Sub-algorithm
Jao, Yoshida [17]	60	3 hours	ρ-method
Izu, Takenaka, Yasuda [15, 16]	83	14 hours	BSGS method
Sakemi et al. [21]	128	131 hours	BSGS method
Sakemi et al. [22]	128	136 hours	$\boldsymbol{\rho}$-method
This paper	**160**	**1314 days**	$\boldsymbol{\rho}$-method

At the beginning of 2000's, bilinear maps were introduced to establish efficient cryptographic schemes with new functions, whose security rely on the infeasibility of newly proposed mathematical problems such as Bilinear Diffie-Hellmann Problem (BDHP) [4], ℓ-Strong Diffie-Hellmann Problem (ℓ-SDHP) [2], ℓ-Bilinear Diffie-Hellmann Inversion Problem (ℓ-BDHIP) [1], ℓ-simplified Strong Diffie-Hellmann Problem (ℓ-sSDHP) [3], and ℓ-BDHEP [5]. In 2006, Cheon defined the discrete logarithm problem with auxiliary input (DLPwAI) as a generarization of some mathematical problems in the above [8]: find α from $G,\ \alpha G,\ \alpha^d G \in \mathbb{G}$ and a positive integer d satisfying $d|(r-1)$. Cheon also proposed a novel algorithm for solving DLPwAI [8, 9]. The time complexity of Cheon's algorithm is $O\left(\sqrt{(r-1)/d} + \sqrt{d}\right)$, and especially when d can be chosen as $d \approx \sqrt{r}$, the complexity becomes $O(\sqrt[4]{r})$, which is more efficient than that for solving DLP in general groups (which requires $O(\sqrt{r})$). Thus, it is indispensable to evaluate the infeasibility of DLPwAI from implementational viewpoints in order to adopt cryptographic schemes based on such new mathematical problems in practice.

In this paper, we investigate useful techniques for speeding up Cheon's algorithm, and demonstrate that it is possible to solve 160-bit DLPwAI over a pairing-friendly elliptic curve within *a practical time*. Specifically, we clarify that Cheon's algorithm effectively works by using some accelerating techniques such as a precomputation table technique effective for scalar multiplications needed for the algorithm, the automorphism technique, and parallelization (see section 3 for details). In fact, we have successfully solved a DLPwAI in 25 days with about 160 cores (1314 days with a single core), which amounts USD 3,150 in Amazon EC2, in a group with 160-bit order defined on the pairing-friendly elliptic curve proposed by Barreto and Naehrig [6].

As far as the authors know, this is the largest result of solving DLPwAI by Cheon's algorithm (see Table 1). Note that solving DLP on this 160-bit elliptic curve is regarded to be infeasible. Our result implies that, if USD 1,000,000 is available, a DLPwAI on the 192-bit Barreto-Naehrig elliptic curve can be solved. Implications of our experimental results to the security of some cryptographic schemes are also discussed in this paper.

Algorithm 1 Cheon's Algorithm [8, 9]

Require: : $G,\ G_1 = \alpha G,\ G_d = \alpha^d G \in \mathbb{G}$, d dividing $r-1$
Ensure: : $\alpha \in \mathbb{Z}/r\mathbb{Z}$
1: Find a generator $\zeta \in (\mathbb{Z}/r\mathbb{Z})^*$
2: Set $\zeta_d \leftarrow \zeta^d$
3: [Step 1] Find $0 \le k_1 < (r-1)/d$ such that $G_d = \zeta_d^{k_1} G$
4: Set $\zeta_e \leftarrow \zeta^{(r-1)/d}$, $G_e \leftarrow \zeta^{-k_1} G_1$
5: [Step 2] Find $0 \le k_2 < d$ such that $G_e = \zeta_e^{k_2} G$
6: Output $\zeta^{k_1 + k_2 (r-1)/d}$

2 Preliminaries

This section introduces Cheon's algorithm for solving DLPwAI [8, 9] and ρ-method [19].

2.1 Cheon's Algorithm

Let $\mathbb{G} = \langle G \rangle$ be an additive cyclic group generated by an element G of prime order $r > 2$. The discrete logarithm problem with auxiliary input (DLPwAI) is a problem to find α on input $G,\ G_1 = \alpha G,\ G_d = \alpha^d G \in \mathbb{G}$ and an integer d dividing $r-1$. In 2006, Cheon proposed a novel algorithm for solving DLPwAI (Cheon's algorithm, [8, 9]), which is the center topic of this paper. Cheon's algorithm requires $O\left(\sqrt{(r-1)/d} + \sqrt{d}\right)$ group operations in time. Especially, when $d \approx \sqrt{r}$, it only requires $O(\sqrt[4]{r})$ operations, which is much smaller than required in the baby-step giant-step (BSGS) method or in the ρ-method for solving DLP.

Let us briefly describe how Cheon's algorithm works. A goal of Cheon's algorithm is to find an integer $k \in \mathbb{Z}/r\mathbb{Z}$ such that $\alpha = \zeta^k$ for a generator ζ of the multiplicative group $(\mathbb{Z}/r\mathbb{Z})^*$ (Note that the generator ζ can be found efficiently). Here, such k is uniquely determined. In order to find k, Cheon's algorithm searches two integers $k_1,\ k_2$ such that $k = k_1 + k_2(r-1)/d$ satisfies $0 \le k_1 < (r-1)/d,\ 0 \le k_2 < d$ in two steps (see Algorithm 1). Step 1 searches an integer k_1 such that $G_d = \zeta_d^{k_1} G$, since k_1 satisfies $\alpha^d = \zeta_d^{k_1}$ for $\zeta_d = \zeta^d$. Similarly, Step 2 searches an integer k_2 such that $G_e = \zeta_e^{k_2} G$, since k_2 satisfies $\alpha = \zeta^{k_1} \zeta_e^{k_2}$ for $\zeta_e = \zeta^{(r-1)/d}$ and $G_e = \zeta^{-k_1} G_1$.

In Cheon's algorithm, searching k_1 (resp. k_2) in Step 1 (resp. Step 2) requires another sub-algorithm. Since these problems are very similar to DLP in the general setting, the baby-step giant-step method [20] or the ρ-method [19] can be used as a sub-algorithm. Since this paper is interested in Cheon's algorithm combined only with the ρ-method, we briefly describe its outline in the next subsection.

2.2 Pollard's ρ-method

Pollard's ρ-method is one of the algorithms for solving DLP [19], which finds a solution α, from $G,\ \alpha G \in \mathbb{G}$ of prime order r, whose time complexity is $O(\sqrt{r})$

because of the birthday paradox. Let us describe the outline in the context of Cheon's algorithm. Especially, since Step 1 and Step 2 of Cheon's algorithm (Algorithm 1) are almost the same, we focus only on Step 1.

The idea of the ρ-method in Step 1 of Cheon's algorithm is to find a collision $F^{(i)}(G_d) = F^{(j)}(G)$ for a given function $F : \mathbb{G} \to \mathbb{G}$, where $F^{(i)}(P) = F(F^{(i-1)}(P))$ and $F^{(0)}(P) = P$. For an efficient evaluation, the function $F(P)$ is desired to be (i) random as possible, and (ii) of the form $F(P) = \zeta^{f(P)}P$ for some function f on $\mathbb{G}$. Such a function F is called a *random-walk function*. In our experiment, we use

$$F(P) : P \mapsto \zeta_d^{f_e(P)}P$$

with a pseudo-random function $f_e : \mathbb{G} \to \mathbb{Z}/e\mathbb{Z}$, where $e = (r-1)/d$ and $\zeta_d = \zeta^d$. By definition, we have

$$F^{(i)}(G_d) = \zeta_d^{\sum_{l=0}^{i-1} f_e(F^{(l)}(G_d))}G_d \quad \text{and} \quad F^{(j)}(G) = \zeta_d^{\sum_{l=0}^{j-1} f_e(F^{(l)}(G))}G.$$

Thus, one can find k_1 by computing

$$k_1 = \sum_{l=0}^{i-1} f_e(F^{(l)}(G_d)) - \sum_{l=0}^{j-1} f_e(F^{(l)}(G)) \bmod (r-1)/d$$

from a collision $F^{(i)}(G_d) = F^{(j)}(G)$. Since the image of the function F has $(r-1)/d$ elements, the time complexity of Step 1 is $O(\sqrt{(r-1)/d})$ (if the KKM method [18] is used, which will be described later).

In the ρ-method, the distinguished element technique [23] reduces the number of elements to be stored. An element which satisfies the specific condition (the least significant 6 bits of an element are zero, for example) is called a distinguished element. With this technique, one has to store elements $F^{(l)}(G_d)$ and $F^{(l)}(G)$ only when they are distinguished elements. Note that there exists a collision on the distinguished elements: in fact, for a collision $F^{(i)}(G_d) = F^{(j)}(G)$, we have $F^{(i+1)}(G_d) = F^{(j+1)}(G)$, $F^{(i+2)}(G_d) = F^{(j+2)}(G)$, $\cdots$, and thus, we eventually have a collision $F^{(i+w)}(G_d) = F^{(j+w)}(G)$ on the distinguished elements for an integer w. The space complexity (also the number of elements) can be reduced to $1/w$ with arbitrary parameter w, while the time complexity is increased to $O\left(\sqrt{(r-1)/d} + w\right)$. However, the increase can be neglected since $w \ll (r-1)/d$ in practice.

3 Implementation

This section describes our strategy for implementing Cheon's algorithm.

3.1 Evaluating $F(X)$

In Cheon's algorithm, the most computationally heavy operation is the evaluation of the function $F^{(l)}(P) = F(F^{(l-1)}(P))$, which consists of

1. Evaluate $f_e(F^{(l-1)}(P))$,
2. Compute $\zeta_d^{f_e(F^{(l-1)}(P))}$ as an exponentiation in $(\mathbb{Z}/r\mathbb{Z})^*$,
3. Compute $\zeta_d^{f_e(F^{(l-1)}(P))}P$ as a scalar multiplication in $\mathbb{G}$.

In our implementation, an element $P \in \mathbb{G}$ is represented by a pair of x-coordinate and y-coordinate, and we used the pseudo-random function $f_e(P) = x(P) \bmod e$, where $x(P)$ is the x-coordinate of P. Thus, procedure 1 is negligible compared to procedure 2 and 3.

Procedure 3 computes a scalar multiplication of a fixed element P independent from l, so that a precomputation table for scalar multiplications is significantly effective (KKM method, [18]). Let us describe the KKM method for a scalar multiplication δP ($\delta \in \mathbb{Z}/r\mathbb{Z},\ P \in \mathbb{G}$). For a fixed integer c (which will be optimized later) and $n = \lceil \sqrt[c]{r} \rceil$, obtain the n-array expansion of the scalar $\delta = \sum_{l=0}^{c-1} \delta_l n^l$ ($0 \leq \delta_l < n$). For all $0 \leq l < c$ and $0 \leq l' < n$, compute $S(l, l') = l' n^l P$ and store them in a table in advance to the scalar multiplications. Then, the scalar multiplication δP is computed by

$$\begin{aligned}\delta P &= \delta_0 P + \delta_1 nP + \cdots + \delta_{c-1} n^{c-1} P \\ &= S(0, \delta_0) + S(1, \delta_1) + \cdots + S(c-1, \delta_{c-1}).\end{aligned}$$

Note that the precomputation table can be computed by at most cn additions.

Similar to procedure 3, procedure 2 also computes an exponentiation of a fixed element ζ_d independent from l, so that the KKM method can be applied to procedure 2 in the same way.

3.2 Using Automorphisms

If there exists an efficiently computable automorphism $\phi : \mathbb{G} \to \mathbb{G}$ of order m on a group $\mathbb{G}$ satisfying the condition $\phi(P) = \zeta_d^s P$ for an integer s, the random-walk function $F(P) : \mathbb{G} \to \mathbb{G}$ can be extended to the random-walk function $\tilde{F}(P) : \mathbb{G}/\sim_\phi \to \mathbb{G}/\sim_\phi$ on the set $\mathbb{G}/\sim_\phi$ of the equivalence classes. Here, two elements $P,\ Q$ are in the same equivalence class if and only if there exists an integer l such that $P = \phi^{(l)}(Q)$ ($0 \leq l < m$). Since the number of elements in $\mathbb{G}/\sim_\phi$ is reduced to $1/m$, the ρ-method can be sped-up by a factor of $\sqrt{m}$.

In our experiment, the pairing-friendly elliptic curve introduced by Barreto-Naehrig (BN curve, [6]) is used. The BN curve is an elliptic curve $y^2 = x^3 + b$ ($b \in \mathbb{F}_p$) defined over a prime field $\mathbb{F}_p$ satisfying $3|(p-1)$. On the BN curve, there exist the negation map ($P \mapsto -P$) which is an automorphism of order 2, and, in addition, the automorphism of order 3 [13]. For an element $P = (x, y) \in \mathbb{G}$, the map $\phi_3(P) = (\epsilon x, y) = \gamma P$ is an automorphism of order 3, where ϵ is a fixed primitive cube root of a unity in $\mathbb{F}_p$ and $\gamma \in (\mathbb{Z}/r\mathbb{Z})^*$ satisfies $\gamma^2 + \gamma + 1 \equiv 0 \bmod r$, i.e. γ is a primitive cube root of unity in $(\mathbb{Z}/r\mathbb{Z})^*$. Such automorphism ϕ_3 can be computed with one multiplication in $\mathbb{F}_p$ only.

Let us consider when these automorphisms satisfy the condition $\phi(P) = \zeta_d^s P$ on the BN curve.

– Negation map: Since $-1 = \zeta^{(r-1)/2} \in (\mathbb{Z}/r\mathbb{Z})^*$, the negation map satisfies the condition if $2d|(r-1)$. The ρ-method can be sped-up by $\sqrt{2}$ with the negation map.
– Automorphism ϕ_3: Since γ is a primitive cubic root of a unity in $(\mathbb{Z}/r\mathbb{Z})^*$, γ can be represented by $\gamma = \zeta^{(r-1)/3}$. Thus, the automorphism satisfies the condition if $3d|(r-1)$. The ρ-method can be sped-up by $\sqrt{3}$ with the automorphism ϕ_3.

As a result of the above analysis, the time complexity of Cheon's algorithm can be reduced to $\tilde{T} = O\left(\sqrt{e/\gcd(d,6)} + \sqrt{d/\gcd(e,6)}\right)$ by using the automorphism technique.

For a random-walk function $\tilde{F}$ of additive type such as Teske's adding walk [24] or the function proposed by [12], the function $\tilde{F}$ on $\mathbb{G}/\sim_\phi$ can fall into short cycles, which are called "fruitless cycles", and hence the optimal speed-up cannot be expected in general [11][12]. However, since our random-walk function is of multiplicative type, our function on $\mathbb{G}/\sim_\phi$ rarely falls into fruitless cycles. Therefore, using both the negation map and the automorphism ϕ_3, the time complexity of the algorithm can be reduced to $\tilde{T} = O\left(\sqrt{e/\gcd(d,6)} + \sqrt{d/\gcd(e,6)}\right)$.

When the above automorphism technique is used, all elements have to be converted to the representative elements of equivalence classes. In our implementation, the representative element is the smallest element when a concatenation $x(P)||y(P)$ is regarded as an integer. Since there are at most 6 elements in an equivalence class, and x-coordinates of a half coincide with those of another half, only one multiplication in $\mathbb{F}_p$ is enough to compute the representative element.

3.3 Parallelization

The ρ-method can be sped-up by parallelization. However, in order to make paths different, initial elements are randomized in the following way [9]: when a core computes $F^{(l)}(G_d),\ F^{(l)}(G)$ for Step 1, two random integers $c_{\mathrm{L}}, c_{\mathrm{R}}$ are assigned to this core and initial points are converted to $G'_d = \zeta_d^{c_{\mathrm{L}}} G_d$ and $G' = \zeta_d^{c_{\mathrm{R}}} G$. Then, one can find k_1 by computing

$$k_1 = \left(\sum_{l=0}^{i-1} f_e(F^{(l)}(G'_d)) + c_{\mathrm{L}}\right) - \left(\sum_{l=0}^{j-1} f_e(F^{(l)}(G') + c_{\mathrm{R}}\right) \bmod (r-1)/d$$

from a collision $F^{(i)}(G'_d) = F^{(j)}(G')$. Note that since all converted initial points can be regarded as scalar multiple point of G, G_d, or G_e, the KKM method can be applied to the conversion.

In our experiment, we also developed a management system for parallelized ρ-method. Outline of the system is described in the appendix.

4 Experimental Results

This section reports our experimental results of Cheon's algorithm for the pairing-friendly elliptic curve with 160-bit order. We have successfully solved a DLPwAI

on this curve in 1314 core days. We emphasize that DLP on the same elliptic curve has been believed to be secure.

4.1 Parameters

We used an additive cyclic group $\mathbb{G}$ with order r on the pairing-friendly elliptic curve $E : y^2 = x^3 + 3$ over a prime field $\mathbb{F}_p$ introduced by Barreto-Naehrig [6]. Concrete values of these parameters are summarized in the following:

$$\begin{aligned}
p &= 146150162449679026514544858992078549371725889081 9 \text{ (160-bit)} \\
\#\mathbb{G} &= 146150162449679026514544738099497118849930002761 3 \text{ (160-bit, prime)} \\
r &= 146150162449679026514544738099497118849930002761 3 \text{ (160-bit)} \\
r - 1 &= 2^2 \cdot 3 \cdot 12132793 \cdot 164442871007 \cdot 448873741399 \cdot 135993458106516349
\end{aligned}$$

where $\#\mathbb{G}$ denotes the number of elements in the additive group $\mathbb{G} = E(\mathbb{F}_p)$. In our implementation, we used the following parameters:

$$\begin{aligned}
d &= 2 \cdot 3 \cdot 12132793 \cdot 135993458106516349 \text{ (84-bit)} \\
e &= (r-1)/d = 2 \cdot 164442871007 \cdot 448873741399 \text{ (77-bit)} \\
\zeta &= 2
\end{aligned}$$

where the generator $\zeta \in (\mathbb{Z}/r\mathbb{Z})^*$ was selected as the smallest one. With these parameters, Step 1 can be sped-up by $\sqrt{2}$, and Step 2 can be sped-up by $\sqrt{6}$ with the automorphism technique, and the estimated time complexity is $2^{40.5}$.

We selected the solution α as 49 decimal places of the circle ratio π:

$$\alpha = 1415926535897932384626433832795028841971693993751 \text{ (160-bit)}$$

We used a base point G whose x-coordinate coincides 48 decimal places of the Napier's constant. Then, coordinates of $G,\ G_1 = \alpha G,\ G_d = \alpha^d G$ are as follows:

$$\begin{aligned}
x(G) &= 718281828459045235360287471352662497757247093699 \\
y(G) &= 267920135876087743710291823125072055976344820822 \\
x(G_1) &= 673981942030616258426617938323441969041367773762 \\
y(G_1) &= 114565531217291633925135194041429741558512233007 2 \\
x(G_d) &= 113217660152885721186991580289363094493274367616 2 \\
y(G_d) &= 948528425611362859774760991656937949436755965122
\end{aligned}$$

With these parameters, we have optimized $n = 2^{20}, c = 8$ for the KKM method. With this optimization, one F-evaluation requires 24 μseconds (on Intel core i7 2.93GHz) while 980 μseconds without KKM.

4.2 Results

This section reports experimental results for solving DLPwAI. Required resources are summarized in Table 2.

Table 2. Required resource for solving a 160-bit DLPwAI

CPU (Hz)	# of PCs	# of cores	Time
Step 1			
Q9450 (2.66GHz)	8	32	7 days
Step 2			
Q9450 (2.66GHz)	8	32	18 days
Q9450 (3.00GHz)	8	32	13 days
X3460 (2.80GHz)	10	80*	1 day
Pentium D (3.40GHz)	9	18	1 day

*Hyper-Threading is used

Step 1 Step 1 required 7 days with 8 PCs (Intel Core2 Quad CPU Q9450 2.66GHz), namely 32 cores: 16 cores are used for computing $F^{(l)}(G'_d)$ while other 16 cores are for computing $F^{(l)}(G')$. The required storage for distinguished elements was 53.3 MByte in total. Obtained partial solution was $k_1 = 108516124982482634887141$.

Step 2 Step 2 was estimated to require 4.7 times more cores compared to Step 1. Thus, we used many PCs with different specifications as in Table 2. Thanks to the flexibility of our management system for the parallelization, PCs are invested one-by-one (see Figure 1). In total, Step 2 required 18 days with 35 PCs (162 cores), more precisely 1090 core days. The required storage for distinguished elements was 256 MByte in total. Obtained partial solution was $k_2 = 60161665500021502744798 50$ and k is obtained by

$$\begin{aligned} k &= k_1 + k_2(r-1)/d \\ &= 888155679312448193339542847931449754121424529241. \end{aligned}$$

Consequently, the final solution α is obtained by

$$\begin{aligned} \alpha &= \zeta^k \bmod r \\ &= 141592653589793238462643383279502884197169399375 1, \end{aligned}$$

which required 1314 core days in total.

4.3 Discussion

Let us estimate the required monetary cost of our experiment on Amazon Elastic Compute Cloud (Amazon EC2), a service to provide resizable computing environment in the cloud. In Amazon EC2, various instances corresponding to CPU power, memory size, and storage size are available. For our experiments, high-spec CPUs and large memory (for the KKM method) are required. Thus, the high-CPU extra-large instance (Memory: 7 GB, Cores: 8 (in virtual)) is adapted,

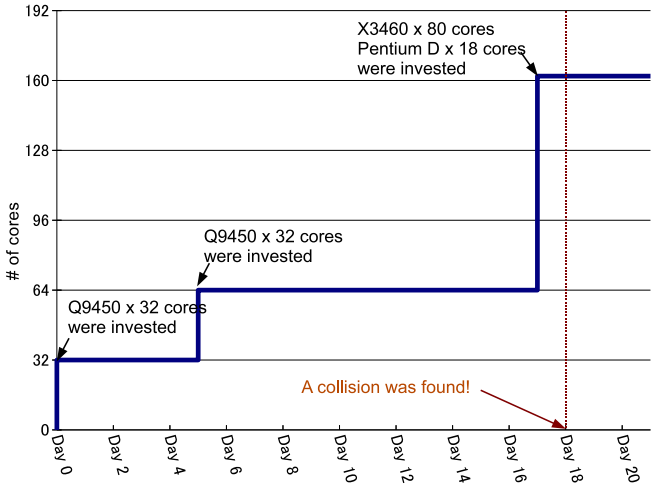

Fig. 1. One-by-one investment in Step 2

which requires USD 0.1 per hour per 1 virtual core. Since our experiment required 1314 core days ($T = 2^{40.5}$), it is estimated to cost USD 3,150 in Amazon EC2. If USD 1,000,000 is available, it is estimated to use 320 times more PCs ($T = 2^{48.3}$) than the experiment. With this environment and if the parameter d can be selected as $d \approx \sqrt[4]{r}$, it is possible to solve a DLPwAI on an elliptic curve with 193.2-bit order.

On the other hand, if USD 1,000,000 is available for the same parameters (namely, a group with 160-bit order), the parameter d can be reduced to $d = 2^{64}$, while d was optimized as $d \approx \sqrt[4]{r}$ in our experiment. Effects of this reduction will be discussed in the next section.

5 Feedback to Cryptographic Schemes

In recently proposed cryptographic schemes, the infeasibility of new mathematical problems are assumed. For example, ℓ-BDHEP is used in Boneh, Gentry, and Waters' broadcast encryption system [5], where ℓ-BDHEP is the problem to find $e(G, \hat{G})^{\alpha^{\ell+1}}$ for a given bilinear map $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$ on input $G,\ \alpha G, \ldots, \alpha^{\ell} G,\ \alpha^{\ell+2} G, \ldots,\ \alpha^{2\ell} G \in \mathbb{G}$ and $\hat{G} \in \hat{\mathbb{G}}$, where $\mathbb{G} = \langle G \rangle$, $\hat{\mathbb{G}} = \langle \hat{G} \rangle$, and $\mathbb{G}_T$ is a multiplicative group with order r. Let d be the largest divisor of $(r-1)$ among $2, 3, \ldots, \ell, \ell+2, \ldots, 2\ell$. As shown in section 4.3, if the parameter d can be selected as $d \approx 2^{80}$ and a 160-bit elliptic curve is used, Cheon's algorithm can solve a DLPwAI. In addition, if USD 1,000,000 is available, the

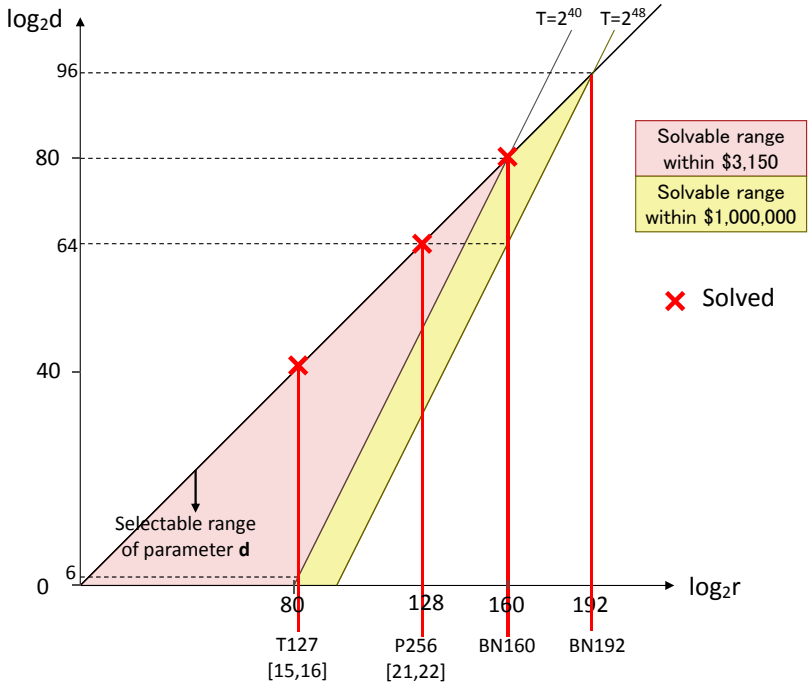

Fig. 2. Selectable range of d

parameter d can be reduced to $d = 2^{64}$ with a 160-bit elliptic curve. Therefore, if the parameter ℓ is chosen to be larger than 2^{80} (or 2^{64}), Cheon's algorithm can solve the ℓ-BDHEP and thus break the scheme: by finding α as a DLPwAI, a solution of ℓ-BDHEP is obtained. Thus, when such cryptographic schemes are implemented with a 160-bit elliptic curve, the parameter ℓ should be smaller than 2^{80} (or 2^{64}).

In this section, we discuss feedbacks of our experiments on a 160-bit elliptic curve to some cryptographic schemes including Boneh, Gentry, and Waters' broadcast encryption scheme [5], Boneh and Boyen's ID-based encryption [1], and Boneh and Boyen's signature scheme [2].

5.1 Boneh, Gentry, and Waters' Broadcast Encryption Scheme

Boneh, Gentry, and Waters' broadcast encryption scheme is provably secure under an assumption that ℓ-BDHEP is infeasible [5], where ℓ is the number of users (receivers) in the broadcast encryption scheme. In the special construction, the sender publishes his public key as

$$\mathsf{pk} = (G, \alpha G, \ldots, \alpha^{\ell} G, \alpha^{\ell+2} G, \ldots, \alpha^{2\ell} G, \gamma G) \in \mathbb{G}^{2\ell+1}$$

where $\gamma \in \mathbb{Z}/r\mathbb{Z}$ is a random number. Thus, when the broadcast encryption scheme is implemented with a 160-bit elliptic curve, ℓ should be chosen smaller than 2^{80} (or 2^{64}) to avoid Cheon's algorithm for DLPwAI.

However, restricting $\ell < 2^{80} \approx 10^{24}$ has almost no effect on the scheme in practice since 10^{24} is far beyond the population on the earth. Even if USD 1,000,000,000 is available, ℓ can be chosen as $2^{64} \approx 10^{19.2}$ so that the restriction has little effect.

5.2 Boneh and Boyen's ID-based Encryption Scheme

Boneh and Boyen's ID-based encryption scheme is proved to be IND-sID-CCA secure under an assumption that ℓ-BDHIP is infeasible [1], where ℓ is the number of queries to the key generation algorithm. Here, ℓ-BDHIP is a problem to find $e(G,G)^{1/\alpha} \in \mathbb{G}_T$ on input $G,\ \alpha G, \ldots, \alpha^{\ell} G \in \mathbb{G}$. Thus, when the ID-based encryption scheme is implemented with a 160-bit elliptic curve, ℓ should be smaller than 2^{80} (or 2^{64}) to avoid Cheon's algorithm for DLPwAI. In the ID-based encryption scheme, queries to the key generation algorithm will be online so that such queries are almost impossible for adversaries. Note that the same discussion can be applied to some ID-based encryption schemes [3, 14].

5.3 Boneh and Boyen's Signature Scheme

Boneh and Boyen's signature scheme is provable secure under the assumption that ℓ-SDHP is infeasible [2] (moreover, it is proven that the infeasibility and the unforgeability is equivalent [17]), where ℓ is the number of queries to the signing algorithm. Here, ℓ-SDHP is the problem to find a pair $(a, \frac{1}{a+\alpha} G) \in \mathbb{Z}/r\mathbb{Z} \times \mathbb{G}$ on input $G,\ \alpha G, \ldots, \alpha^{\ell} G \in \mathbb{G}$ and $\hat{G} \in \hat{\mathbb{G}}$. Thus, when the signature scheme is implemented with a 160-bit elliptic curve, ℓ should be smaller than 2^{80} (or 2^{64}) to avoid Cheon's algorithm for DLPwAI. The effect of this restriction depends on how the signing algorithm is implemented. If it is implemented online similar to Boneh and Boyen's ID-based encryption scheme, this restriction has almost no effect. However, if the query to the signing algorithm can be offline (for example, the case where the signing algorithm is implemented in IC chip), more queries will be available compared to the online case. Thus, this case is the most attackable for adversaries with Cheon's algorithm.

6 Concluding Remarks

This paper successfully solved a discrete logarithm problem with auxiliary input (DLPwAI) in 1314 core days over a 160-bit pairing-friendly elliptic curve. If cryptographic schemes based on mathematical problems such as ℓ-BDEP, ℓ-SDHP, ℓ-sSDHP, or ℓ-BDHIP are implemented, such weak parameters should be avoided. However, there are pairing-based cryptographic schemes which are not affected by Cheon's algorithm such as Boneh and Franklin's ID-based encryption scheme.

Acknowledgements

We wish to thank anonymous reviewers for their helpful comments and suggestions. We also thank Professor Yoshitaka Morikawa, Professor Yasuyuki Nogami, and Naoki Takahashi for their generous support.

References

1. D. Boneh and X. Boyen, "Efficient Selective-ID Secure Identity-based Encryption without Random Oracles", *EUROCRYPT 2004*, LNCS 3027, pp. 223-238, Springer, 2004.
2. D. Boneh and X. Boyen, "Short Signatures without Random Oracles", *EUROCRYPT 2004*, LNCS 3027, pp. 56-73, Springer, 2004.
3. D. Boneh, X. Boyen, and E. Goh, "Hierarchal Identity Based Encryption with Constant Size Ciphertext", *EUROCRYPT 2005*, LNCS 3494, pp. 440-456, Springer, 2005.
4. D. Boneh and M. Franklin, "Identity-based Encryption from the Weil Pairing", *CRYPTO 2001*, LNCS 2139, pp. 213-229, Springer, 2001.
5. D. Boneh, C. Gentry, and B. Waters, "Collusion Resistant Broadcast Encryption with Short Ciphertext and Private Keys", *CRYPTO 2005*, LNCS 3621, pp. 258-275, Springer, 2005.
6. P. Barreto and M. Naehrig, "Pairing-friendly Elliptic Curves of Prime Order", *SAC 2005*, LNCS 3897, pp. 319-331, Springer, 2006.
7. J.W. Bos, M.E. Kaihara, T. Kleinjung, A.K. Lenstra, and P.L. Montgomery, "PlayStation 3 Computing Breaks 2^{60} Barrier 112-bit Prime ECDLP Solved", 2009. `http://lacal.epfl.ch/112bit_prime`
8. J.H. Cheon, "Security Analysis of Strong Diffie-Hellman Problem", *EUROCRYPT 2006*, LNCS 4004, pp. 1-11, Springer, 2006.
9. J.H. Cheon, "Discrete Logarithm Problems with Auxiliary Inputs", *Journal of Cryptology*, vol. 23, no. 3, pp. 457-476, 2010.
10. Distributed.net, `http://www.distributed.net/Main_Page/`
11. S. D. Galbraith and R. S. Ruprail, "Using Equivalence Classes to Accelerate Solving the Discrete Logarithm Problem in a Short Interval", *PKC 2010*, LNCS 6056, pp. 368-386, Springer, 2009.
12. R. Gallant, R. Lambert, and S. Vanstone, "Improving the Parallelized Pollard Lambda Search on Binary Anomalous Curves", *Math. Comp.*, vol. 69, pp. 1699-1705, 2000.

13. R. Gallant, R. Lambert, and S. Vanstone, "Faster Point Multiplication on Elliptic Curves with Efficient Endomorphisms", *CRYPTO 2001*, LNCS 2139, pp. 190-200, Springer, 2001.
14. C. Gentry, "Practical Identity-based Encryption without Random Oracles", *EUROCRYPT 2006*, LNCS 4004, pp. 445-464, Springer, 2006.
15. T. Izu, M. Takenaka, M. Yasuda, "Experimental Results on Cheon's Algorithm", *WAIS 2010*, pp. 625-630 in the proceedings of *ARES 2010*, IEEE Computer Science, 2010.
16. T. Izu, M. Takenaka, M. Yasuda, "Experimental Analysis of Cheon's Algorithm against Pairing-friendly Curves", Journal of Information Processing vol. 19, pp. 441-450, 2011.
17. D. Jao and K Yoshida, "Boneh-Boyen Signatures and the Strong Diffie-Hellman Problem", *Pairing 2009*, LNCS 5671, pp. 1-16, Springer, 2009.
18. S. Kozaki, T. Kutsuma, and K. Matsuo, "Remarks on Cheon's Algorithms for Pairing-related Problems", *Pairing 2007*, LNCS 4575, pp. 302-316, Springer, 2007.
19. J. Pollard, "Monte Carlo Methods for Index Computation (mod p)", *Math. Comp.*, vol. 32, pp. 918-924, 1978.
20. D. Shanks, "Class Number, a Theory of Factorization, and Genera", *Proc. of Symp. Math. Soc.*, vol. 20, pp. 41-440, 1971.
21. Y. Sakemi, T. Izu, M. Takenaka, and M. Yasuda, "Solving DLP with Auxiliary Input over an Elliptic Curve Used in TinyTate Library", *WISTP 2011*, LNCS 6633, pp. 116-127, Springer, 2011.
22. Y. Sakemi, T. Izu, M. Takenaka, and M. Yasuda, "Solving a DLP with Auxiliary Input with the ρ-algorithm", *WISA 2011*, LNCS 7115, pp. 98-107, Springer, 2012.
23. E. Teske, "Speeding up Pollard's Rho Method for Computing Discrete Logarithms", *ANTS III*, LNCS 1423, pp. 541-554, Springer, 1998.
24. E. Teske, "On Random Walks for Pollard's Rho Method", *Math. Comp.*, vol. 70, pp. 809-825, 2001.

A Large Scale Solving System

This appendix describes the management system "Large Scale Solving System (LSSS)" for the parallelized ρ-method for Cheon's algorithm dedicated to the large-scale experiment.

For such a large-scale parallelized experiment, the distributed.net is used worldwide [10], which supports to solve large scale problems using idle PCs, CPUs or GPUs in everywhere in the world. For example, the distributed.net has broken RC5-64 (64-bit RC5), and is trying to break RC5-72 (72-bit RC5) currently. Anyone can join to the distributed.net simply by downloading and executing a client program. A server of the distributed.net system distributes a "key-block" to each client, and each client exhaustively searches the correct key. In the distributed.net system, the HTTP protocol over proxy-server is used for the communication between the server and clients.

the distributed.net has high scalability and is suitable for large-scale experiment. However, since the connection between the server and clients in the distributed.bet is loose, the system is not efficient. Thus, we have designed more tightly-connected and more efficient but less scalable system in our experiment.

An overall design of LSSS is shown in Figure 3. We have also adopted the HTTP protocol over proxy-server for the communication between the server and clients so that any clients of any organizations can join to LSSS at any time (this is very important when the experiments are conducted in academic organizations and private companies).

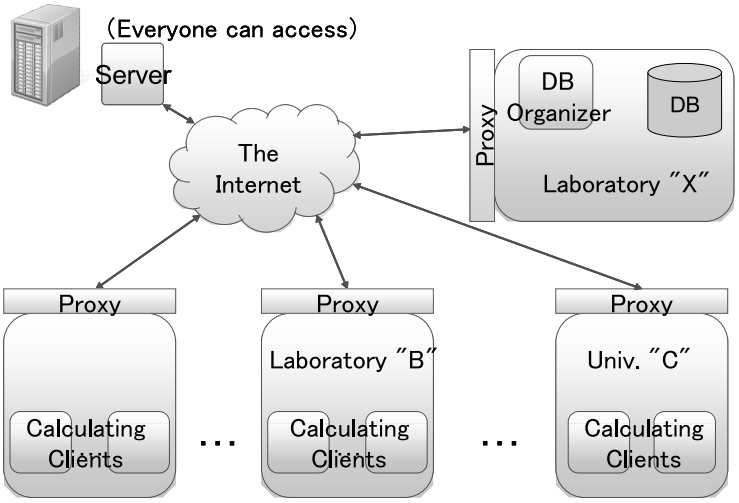

Fig. 3. Grand design of LSSS

For solving a 160-bit DLPwAI by Cheon's algorithm in our experiment, two LSSSs are used as in Figure 4. Each LSSS consists of one server, numerous calculating clients, and one DB organizer. A calculating client evaluates the random-walk function and outputs a result if it is the distinguished element. Every client sends distinguished elements to the server and the server catches the received distinguished elements. A DB organizer obtains the distinguished elements from the server and establishes a DB of these distinguished elements. One LSSS is dedicated to evaluate $F^{(l)}(G'_d)$, while another LSSS is to evaluate $F^{(l)}(G')$ for Step 1 of Cheon's algorithm. These two DBs are compared by a DB matching tool periodically. If a collision $F^{(i)}(G'_d) = F^{(j)}(G')$ is found in these DBs, the tool output the collision. In LSSS, all functions work on Windows and Linux (and perhaps other UNIX OSs) to utilize any platforms.

A calculating client has the common communication unit and the calculating unit. Since the common communication unit is independent from the target parameter, and APIs between the communication unit and the calculating unit is very simple, a user has to change the calculating unit only for a new experiment. Because of this construction, LSSS can be used not only for Cheon's algorithm but also for solving ECDLP with ρ-method and other similar problems.

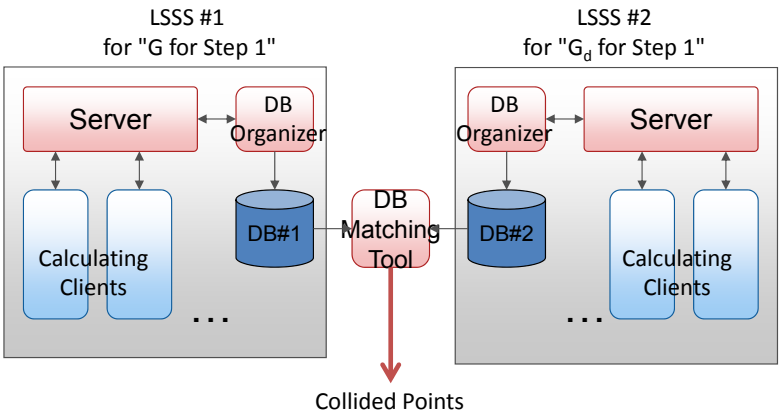

Fig. 4. Constitution of the solving system of a 160-bit DLPwAI by Cheon's algorithm with two LSSSs